Implementace zásobníku v Pythonu

List vs. Deque: Co se děje v paměti?

Informatický seminář

13. února 2026

Rekurze vs. Vlastní zásobník

Proč nahrazovat rekurzi (DFS) vlastním cyklem?

Rekurze:

+ Jednoduchý, čitelný kód.
- Limitovaná hloubka (`RecursionError`).
- Režie volání funkcí (stack frames).

Vlastní zásobník (Iterace):

+ Neomezená hloubka (limitováno jen RAM).
+ Často rychlejší.
- Složitější správa stavu.

Kandidát 1: Python `list`

Co to je? Dynamické pole (Dynamic Array).

Jak funguje v paměti

- Data jsou uložena v jednom souvislém bloku paměti.
- **Append/Pop na konci:** Velmi rychlé $O(1)$.
- **Nevýhoda:** Když dojde místo, Python musí alokovat větší pole a vše zkopírovat (Amortizovaná složitost).
- **Nebezpečí:** `pop(0)` (zepředu) je extrémně pomalé $O(N)$, protože se všechna data musí posunout!

Kandidát 2: `collections.deque`

Co to je? Doubly Ended Queue (Obousměrná fronta).

Jak funguje v paměti

- Implementováno jako **spojový seznam bloků** (doubly linked list of arrays).
- Data nejsou souvislá, jsou rozsekána do bloků propojených ukazateli.
- **Výhoda:** Garantované $O(1)$ na obou koncích. Žádné velké kopírování při růstu.
- **Nevýhoda:** Mírně vyšší paměťová režie pro malé množství dat.

Srovnání v kódu

List (Zásobník)

```
stack = []

# Push
stack.append(x)

# Pop (LIFO)
if stack:
    x = stack.pop()
```

Deque (Zásobník)

```
from collections import deque
stack = deque()

# Push
stack.append(x)

# Pop (LIFO)
if stack:
    x = stack.pop()
```

Syntaxe je téměř totožná. Rozdíl je ve vnitřním chování.

Kdy co použít?

Použití	Doporučení	Důvod
Zásobník (LIFO)	`list`	Rychlý, méně režie, standardní.
Fronta (FIFO)	`deque`	`list.pop(0)` je $O(N)$! `deque` je $O(1)$.
Oboustranné	`deque`	Efektivní přidávání/odebírání na obou koncích.
Indexování	`list`	Přístup `a[i]` je $O(1)$, u deque $O(N)$.

Závěr pro naši úlohu

Jelikož děláme DFS (prohledávání do hloubky), chováme se jako **Zásobník**. Rozdíl v rychlosti bude minimální, ale deque je robustnější řešení, pokud bychom algoritmus měnili na BFS.